TECHNICAL REPORT

by

Neha Pramanick

October 31, 2025

Contents

1 Introduction

"Security is not a product, but a process."
– **Bruce Schneier**

From Caesar's shift cipher to modern neural distinguishers, encryption has evolved not as a sequence of inventions but as a continuous dialogue — between those who create secrets and those who learn to reveal them. Each era in this dialogue mirrors its technology: mechanical ciphers gave way to digital block ciphers, and now, as algorithms learn to think, machines themselves learn to break them. This report explores that evolution with a focus on the lightweight block cipher **SPECK 32/64**, highlighting both existing machine learning (ML) based cryptanalysis and a proposed approach.

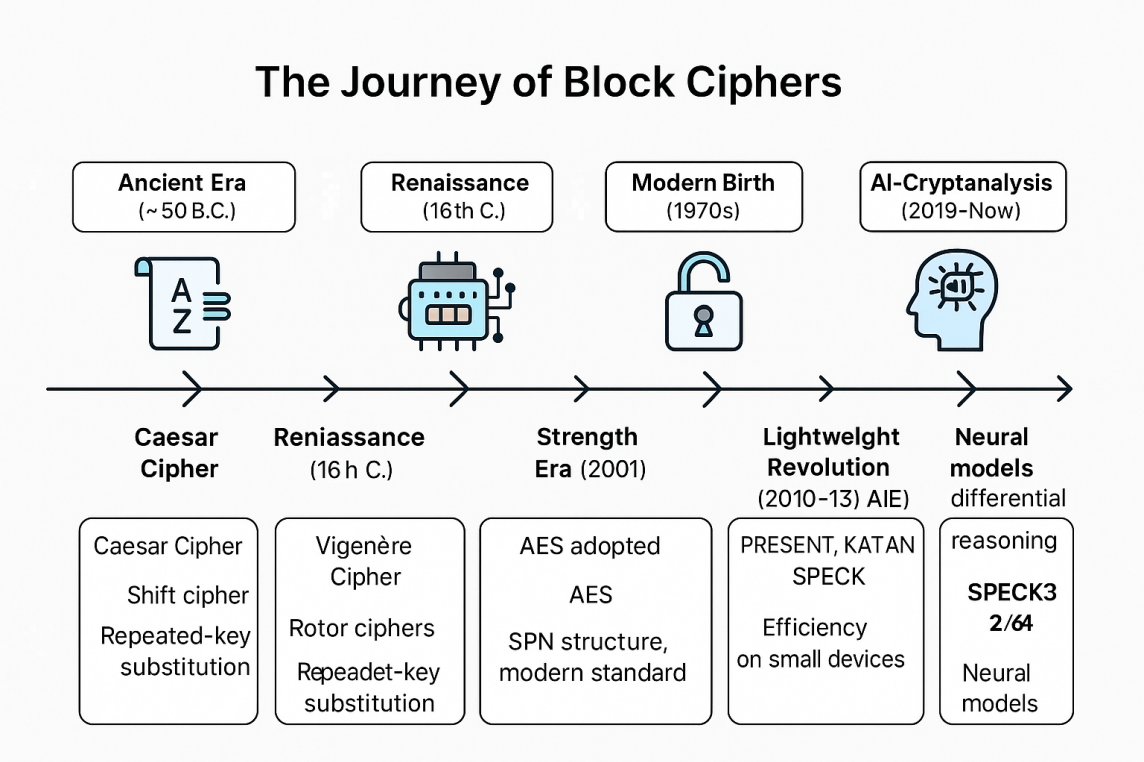

Figure 1: The Journey of Block Ciphers from Ancient Era to ML-based Cryptanalysis.

1.1 Evolution of Block Ciphers

- In the 1970s, the first modern block cipher called **DES** was developed to protect digital communication.
- In the 2000s, **AES** replaced DES and became the global standard for secure encryption.

- In the 2010s, with the rise of IoT and small electronic devices, researchers designed lightweight ciphers such as **PRESENT**, **SIMON**, and **SPECK** to provide strong security with low power and memory use.
- In recent years (2019–now), new studies have used ML to understand and attack block ciphers. Neural networks are now helping researchers detect hidden patterns that traditional analysis might miss.

1.2 Block Cipher Speck 32/64

- The **SPECK** family of lightweight block ciphers was developed by the **U.S. National Security Agency (NSA)** and first introduced publicly in 2013 by **Beaulieu et al.** in their paper [1].
- The design goal of SPECK was to create a software-friendly lightweight cipher that could achieve high performance on small, resource-constrained devices such as microcontrollers and IoT hardware, while maintaining adequate cryptographic security.
- SPECK follows an **ARX** design — it uses only three basic operations: modular addition, bit rotation, and bitwise XOR. These operations are fast and available on almost all processors. For SPECK 32/64:
  - Block size = 32 bits (two 16-bit words)
  - Key size = 64 bits (four 16-bit words)
  - Number of rounds = 22

1.2.1 Mathematical Model of SPECK 32/64

Let x, y be the two 16-bit words of plaintext and k_i the round key at round i.

Round Function:

$$x' = (\mathrm{ROR}(x, 7) \boxplus y) \oplus k_i,$$
$$y' = \mathrm{ROL}(y, 2) \oplus x'$$

where:

- $\mathrm{ROR}(x, r)$ is a right rotation of x by r bits,
- $\mathrm{ROL}(y, r)$ is a left rotation of y by r bits,
- $\boxplus$ represents addition modulo 2^{16},

- $\oplus$ denotes bitwise XOR.

Key Schedule: The key schedule expands the four 16-bit key words (k_0, l_0, l_1, l_2) to generate 22 round keys using the following recursive equations:

$$l_{i+3} = (\mathrm{ROR}(l_i, 7) \boxplus k_i) \oplus i,$$
$$k_{i+1} = \mathrm{ROL}(k_i, 2) \oplus l_{i+3}.$$

Test Vector (from reference design):

Plaintext: $0x6574\,694C$, Key: $0x1918\,1110\,0908\,0100$, Ciphertext: $0xA868\,42F2$

1.3 Cryptanalysis of Speck 32/64

- The first known cryptanalysis of SPECK 32/64 was presented by E. Abed et al. (2014) [2], who applied traditional differential cryptanalysis to round-reduced versions of the cipher and identified efficient differential characteristics useful for key recovery.
- Later, in 2019, the use of machine learning for cryptanalysis was introduced by M. Gohr, who presented a deep learning-based differential attack on round-reduced SPECK 32/64 at the CRYPTO 2019 conference, which showed improved performance over traditional differential distinguishers for 11 round counts.

2 Machine Learning–Based Cryptanalysis of SPECK 32/64

Overview

The study by M. Gohr (2019) [3] introduced a data-driven approach to differential cryptanalysis, replacing traditional probabilistic modelling with a neural network-based distinguisher.

Analytical Flow

Input Difference $\rightarrow$ Cipher Rounds $\rightarrow$ Ciphertext Distribution
$\rightarrow$ Neural Distinguisher $\rightarrow$ Key Search

Research Framework

Step	Methodology	Key Observation	Result
1	Markov Model Analysis	Examined how a single input difference propagates across 8 rounds.	Characterized differential diffusion in SPECK.
2	Deep Neural Distinguisher	Trained a residual network to classify ciphertext pairs.	Achieved fivefold improvement in mean key rank over classical distinguishers.
3	Bayesian Key Search	Applied Bayesian optimization for adaptive key ranking.	Reduced key search complexity for 11 rounds to approximately 2^{38} encryptions.
4	Feature Identification	Neural model detected dependencies invisible to traditional differential analysis.	Revealed latent statistical structure in ciphertext distributions.

Notable Contributions

- Established a transition from rule-based to data-driven cryptanalysis.
- Modeled neural networks as probabilistic estimators of cipher structure.
- Introduced Bayesian optimization for efficient key search guidance.
- Demonstrated that neural distinguisher can pick the hidden feature of ciphertext distribution that traditional differential methods can't see.

Comparative Framework

Classical Model: Difference $\rightarrow$ Table $\rightarrow$ Probability $\rightarrow$ Distinguisher

$\Downarrow$

Machine Learning Model: Difference $\rightarrow$ Ciphertexts $\rightarrow$ Network $\rightarrow$ Learned Distinguisher

2.1 Mathematical Model and Network Structure (Gohr, 2019)

The following mathematical formulation summarizes and formalizes the training setup and neural network structure described in Section 4 of Gohr (2019). While the original paper expresses these concepts descriptively, the equations below restate them explicitly for analytical clarity.

In Gohr's setting, a neural network is trained to distinguish ciphertext pairs produced by a reduced-round version of the cipher from pairs of random 64-bit strings. Each example consists of a pair of ciphertexts (c_0, c_1) generated from plaintexts (p_0, p_1) with a fixed input difference:

$$p_0 \oplus p_1 = \Delta_{in}.$$

After encryption under the same key k,

$$c_0 = E_k(p_0), \qquad c_1 = E_k(p_1).$$

Pairs generated by the cipher are labeled $y = 1$, while random pairs are labeled $y = 0$. The neural distinguisher ND_θ with parameters θ approximates the posterior probability:

$$ND_\theta(c_0, c_1) \approx \Pr[y = 1 \mid (c_0, c_1)].$$

Training Objective. The network is trained using the mean-squared error loss with L_2 regularization as described by Gohr:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (ND_\theta(x_i) - y_i)^2 + c\|\theta\|_2^2, \tag{1}$$

where $c = 10^{-5}$. Optimization is performed with the Adam optimizer. The batch size is 5000, total training samples 10^7, and up to 200 epochs.

A cyclic learning rate schedule is applied:

$$l_i = \alpha + \frac{(n - i) \bmod (n + 1)}{n}(\beta - \alpha), \tag{2}$$

with parameters $\alpha = 10^{-4}$, $\beta = 2 \times 10^{-3}$, and $n = 9$.

Network Structure. The distinguisher operates on a bit-sliced representation of the ciphertext pair, arranged as a 4×16 binary matrix.

1. **Initial convolution:** A one-dimensional convolution layer with kernel size 1 and 32 output channels, followed by batch normalization and ReLU activation.

2. **Residual tower:** A stack of T residual blocks (typically $T = 10$ for N5/N6, $T = 1$ for N7/N8). Each block applies two convolutional layers (kernel size 3, 32 filters) with batch normalization and ReLU activation, and adds the input to the output:

$$y = x + \mathrm{ReLU}(\mathrm{BN}(W_2 * \mathrm{ReLU}(\mathrm{BN}(W_1 * x)))). \tag{3}$$

3. **Prediction head:** After the residual stack, the feature map is flattened and passed through two fully-connected layers with 64 ReLU units each, followed by a single sigmoid neuron that outputs a probability in $(0, 1)$.

Score Computation. During evaluation, the network output is transformed into a log-likelihood ratio as defined in Gohr's analysis:

$$s_i = \log \frac{ND_\theta(c_0, c_1)}{1 - ND_\theta(c_0, c_1)}. \tag{4}$$

For N independent ciphertext pairs, the total score is accumulated as:

$$S = \sum_{i=1}^{N} s_i. \tag{5}$$

A positive S indicates that the ciphertext pairs are likely to originate from the reduced-round cipher; a negative value indicates random data.

Equations (1)–(5) represent a data-driven generalization of the classical differential probability model. Instead of computing explicit transition probabilities $P(\Delta_{in} \to \Delta_{out})$, the neural distinguisher ND_θ learns to approximate these dependencies directly from data, capturing statistical biases in ciphertext distributions that arise from modular addition and bit rotation operations in the cipher.

3 Related Work

Machine learning-based cryptanalysis has evolved rapidly over the past few years, extending beyond empirical attacks to interpretable and theoretically grounded models. Table 1 summarizes five key papers from 2021–2025 that form the foundation for the present study.

4 Motivation

- Existing optimization-based cryptanalysis frameworks rely on predefined or locally tuned parameters, which restrict their ability to achieve globally optimal configurations.
- Local search strategies often converge prematurely, leaving potential performance improvements unexplored.
- The influence of the number of candidate keys retained per iteration on key recovery performance remains uncertain.
- There is a need for a global, adaptive parameter search mechanism that can automatically identify efficient configurations while maintaining computational balance and interpretability.

5 Objective

- To design a global parameter optimization framework for learning-assisted key search.
- To analyze the relationship between the number of retained candidate keys per iteration and overall cryptanalytic performance.
- To enhance the efficiency, robustness, and adaptability of key search methods through data-driven parameter tuning.

6 Proposed Method

During the initial stage of this work, several existing cryptanalytic models were implemented and analyzed to observe their learning behavior on reduced-round versions of lightweight block ciphers. Although these approaches provided important insights, the obtained results were not sufficiently promising in terms of accuracy and computational efficiency. This motivated a deeper investigation into the core limitations of current learning-based cryptanalytic frameworks, particularly the inefficiencies observed in parameter tuning and candidate key selection.

In the proposed approach, the focus shifts from static or locally tuned parameter configurations towards a more adaptive and self-correcting strategy based on **Bayesian Reinforcement Learning (BRL)**. Instead of relying on precomputed cutoffs or heuristic

local searches, the cryptanalytic process is modeled as an interactive learning environment where an agent continuously refines its belief about the most probable subkey candidates.

Here, the agent operates under the Bayesian framework while interacting with the neural distinguisher. Each subkey hypothesis represents an action, and the distinguisher score acts as the feedback or reward signal. Based on this reward, the BRL agent updates its posterior belief about the likelihood of each subkey being correct. Over successive iterations, the agent learns an optimal exploration policy that balances between trying new candidates (exploration) and reinforcing high-confidence subkeys (exploitation). This eliminates the need for manual parameter tuning and allows the search process to naturally converge toward globally efficient key candidates.

The key idea of this model is to integrate the interpretability of Bayesian reasoning with the adaptivity of reinforcement learning (RL) to achieve a dynamic, feedback-driven key search mechanism. Such an approach is expected to enhance both the robustness and efficiency of cryptanalysis by reducing redundancy in parameter selection and improving convergence behavior across rounds. Let an RL agent decide, online and adaptively, how many top subkey candidates to test at each iteration (and related parameters) using Bayesian posteriors + neural distinguisher feedback, so the key-search becomes adaptive, efficient and less dependent on offline tuning.

6.1 Proposed Model Architecture

The proposed model consists of five main modules as illustrated below:

1. **Data and Neural Distinguisher (ND) Module**
   - **Input:** plaintext–ciphertext pairs or synthetic side-channel traces.
   - The ND, pre-trained on round-reduced versions of the cipher (e.g., SPECK 32/64), estimates the likelihood that given ciphertext pairs originate from the true cipher rather than random permutations.
   - The ND outputs act as probabilistic evidence for Bayesian analysis.
2. **Bayesian Probability Estimation Module**
   - Computes posterior probabilities for each candidate subkey k_i using:

$$P(k_i|L) \propto P(L|k_i)\, P(k_i)$$

     where L represents leakage or ciphertext evidence and $P(L|k_i)$ is estimated from ND outputs.

- These probabilities form the belief distribution used for further decision-making.

3. **Feature Extraction Module**

   - Summarizes the current state of the search process into a feature vector consisting of:
     - Shannon entropy of the posterior distribution,
     - Maximum posterior probability,
     - Number of candidates above a confidence threshold,
     - Estimated noise level or uncertainty, and
     - Remaining computational budget.
   - The resulting feature vector represents the environment state s_t.

4. **Bayesian Reinforcement Learning (BRL) Agent**

   - Receives the state vector s_t and outputs an action a_t, representing the number of candidates to test or the cutoff threshold to apply in the current iteration.
   - The agent aims to balance exploration (testing new candidates) and exploitation (focusing on promising ones).
   - Reward feedback is derived from improvement in posterior confidence or success probability, penalized by computational cost.
   - Through repeated interaction, the BRL agent learns a policy $\pi_\theta(a|s)$ that approximates optimal parameter decisions.

5. **Key Evaluation and Update Module**

   - Executes the candidate testing phase using parameters provided by the BRL agent.
   - Evaluates candidates via partial decryption and ND scoring.
   - Updates the posterior distribution based on outcomes and sends feedback to the BRL agent, forming a closed learning loop.

7 Future Work and Target Submission

The proposed RL-BKS model is currently at the conceptual stage. Next, the plan is to validate the framework through small-scale experiments and integrate the reinforcement learning module with the Bayesian key search process. The final objective is to break more rounds of SPECK 32/64 than current state-of-the-art for submitting to the **IACR Asiacrypt 2026** conference, expected around **May 2026**.

References

[1] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "Simon and speck: Block ciphers for the internet of things," *Cryptology ePrint Archive*, 2015.

[2] F. Abed, E. List, S. Lucks, and J. Wenzel, "Differential cryptanalysis of round-reduced simon and speck," in *International Workshop on Fast Software Encryption*, pp. 525–545, Springer, 2014.

[3] A. Gohr, "Improving attacks on round-reduced speck32/64 using deep learning," in *Annual International Cryptology Conference*, pp. 150–179, Springer, 2019.

[4] A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A deeper look at machine learning-based cryptanalysis," in *Annual international conference on the theory and applications of cryptographic techniques*, pp. 805–835, Springer, 2021.

[5] Z. Bao, J. Lu, Y. Yao, and L. Zhang, "More insight on deep learning-aided cryptanalysis," in *International conference on the theory and application of cryptology and information security*, pp. 436–467, Springer, 2023.

[6] A. Shafran, E. Malach, T. Ristenpart, G. Segev, and S. Tessaro, "Is ml-based cryptanalysis inherently limited? simulating cryptographic adversaries via gradient-based methods," in *Annual International Cryptology Conference*, pp. 37–71, Springer, 2024.

[7] Q. Ling, T. Cui, H. Hu, S. Gong, Z. He, J. Huang, and J. Xiao, "Finding impossible differentials in arx ciphers under weak keys," *IACR Transactions on Symmetric Cryptology*, vol. 2024, no. 1, pp. 326–356, 2024.

[8] Z. Hou, Z. Bao, J. Lu, and S. Chen, "Observations on the bayesiankeysearch with applications to simon and simeck," *IACR Transactions on Symmetric Cryptology*, vol. 2025, no. 3, pp. 755–799, 2025.

Table 1: Summary of Key Works in Machine Learning-based Cryptanalysis (2021–2025)

Authors	**Year**	**Venue**	**Work Done**
A. Benamira et al. [4]	2021	EUROCRYPT	Performed the first **interpretability analysis** of Gohr's neural distinguisher. Showed that deep networks effectively learn the cipher's **Differential Distribution Table (DDT)** and primarily exploit the differential behavior of penultimate and antepenultimate rounds. Developed an analytical distinguisher achieving comparable performance to the neural one.
Z. Bao et al. [5]	2023	ASIACRYPT	Extended neural cryptanalysis by identifying **explicit statistical correlations** between ciphertext partial values and intermediate differences. Proposed training improvements (layer freezing and mini-batch scheduling) and achieved a **14-round related-key recovery attack** on SPECK 32/64, surpassing earlier limits.
A. Shafran et al. [6]	2024	CRYPTO	Provided a formal **theoretical framework** linking classical and gradient-based adversaries. Proved that machine learning attacks can simulate sample-based attacks with bounded overhead, showing ML adversaries are **provably as strong** as traditional probabilistic ones.
Q. Ling et al. [7]	2024	IACR TOSC	Proposed framework for discovering impossible differentials in ARX ciphers under weak keys. The authors analyzed the XOR difference propagation through multiple modular additions and established accurate propagation rules for such constructions. Using these, they identified new **impossible differential (ID) properties** in several lightweight ciphers.
Z. Hou et al. [8]	2025	LATINCRYPT	Refined Gohr's **BayesianKeySearch (BKS)** algorithm via parameter tuning and cutoff optimization. Demonstrated improved key-ranking efficiency and success probability for **16–18 round key recovery** on Simon and Simeck ciphers, making ML-based key search significantly faster and more stable.